theless, the PDCP may feed a single RLC and may fill a single transport block. In some implementations, the use of the indication may address legacy issues, such as when legacy networks omit the indication. Moreover, a local area radio network element may be configured to detect this indication (e.g., as a bit in the PDCP header) and determine whether the PDCP PDU is targeted to a processing entity of the first network (e.g., targeted to its own processing entity, such as the local area RRC entity) or whether it is targeted to a processing entity of the second network (e.g., EPS RRC). As such, the user equipment may determine, when composing and assembling the message into a PDCP PDU, whether to set this indication to the value of the selected routing option.

[0052] FIG. **7**B depicts an example message, in accordance with some example implementations. In the example of FIG. **7**B, the PDCP header field **799** may have a value of "0", which may be defined to indicate legacy compatibility that is to indicate an EPS RRC message, while a value of "1" may be defined to indicate that the RRC message is targeted to the local area network (example.g., to a local area RRC entity). In some implementations, the value "1" may not be understood by some legacy networks, although this may not cause confusion in those legacy networks. Any confusion may be minimized, if not avoided, by for example handling of the PDCP PDU in the local area network node. As such, any RLC PDU with the flag set to "1" may be processed inside the local area network, but it will not transfer to the EPS network. On the other hand, any PDCP PDU with the flag set to "0" may be transferred to the EPS network, where it is then legacy compatible as being the EPS PDCP PDU with its RFU flag set to "0" and carrying EPS RRC message.

[0053] Table 1 depicts an example RRC message definition.

TABLE 1

```
Message ::= SEQUENCE {
    Message DL__DCCH-MessageType
}
    DL__DCCH-MessageType ::= CHOICE {
        C1
        Message__name {
            IE
            IE
    }
    HASH
}
```

[0054] Table 2 depicts an example of protocol encapsulation (e.g., an eRRC.message carries a container including a RRC-message).

TABLE 2

```
Message ::= SEQUENCE {
    Message LTEHi-eMessageType
}
    LTEHi-eMessageType ::= CHOICE {
        C1
        Message__name {
            eRRC__TransactionIdentfier
            eIE
            eIE
            eIE
            DL-DCCH-message {
MessageType
RRC__TransactionIdentifier
IE
```

TABLE 2-continued

```
IE
        }
        HASH
    }
    eHASH
}
```

[0055] Table 3 depicts an alternative approach of a flat protocol implementation (e.g., eRRC-message and RRC-message are separated by MessageType and their processes may be separated by RRC_TransactionIdentifier).

TABLE 3

```
Message ::= SEQUENCE {
    Message LTEHi-eMessageType
}
    LTEHi-eMessageType ::= CHOICE {
        C1
        Message__name {
            eRRC__TransactionIdentifier
            eIE
            eIE
            eIE
    }
    eHASH
}
Message ::= SEQUENCE {
    Message DL-DCCH-MessageType
}
    DL-DCCH-MessageType ::= CHOICE {
        C1
        Message__name {
            RRC__TransactionIdentifier
            IE
            IE
    }
    HASH
}
```

[0056] FIG. **8** depicts an example implementation of an access point **800**, which may be implemented at devices **110**A or **110**B. The access point may include one or more antennas **820** configured to transmit downlinks and configured to receive uplinks via the antenna(s) **820**. The access point may further include a plurality of radio interfaces **840** coupled to the antenna **820**. The radio interfaces may correspond to a plurality of radio access technologies including one or more of LTE, WLAN, Bluetooth, BT-LE, NFC, radio frequency identifier (RFID), ultrawideband (UWB), ZigBee, and the like. The access point may further include at least one processor, such as processor **830**, for controlling the access point **800** and for accessing and executing program code stored in memory **835**. In some example embodiments, the memory **835** includes code, which when executed by at least one processor causes one or more of the operations described herein with respect to an access point (e.g., establishing bearers, establishing RRC entities in the network to communicate with the user equipment's RRC entities, generate RRC signaling/messaging, and the like). The radio interface **840** may further include other components, such as filters, converters (e.g., digital-to-analog converters and the like), mappers, a Fast Fourier Transform (FFT) module, and the like, to generate symbols for a transmission via one or more downlinks and to receive symbols (e.g., via an uplink).